Transfer Learning

Lisa Torrey and Jude Shavlik

University of Wisconsin, Madison WI, USA

Abstract. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. While most machine learning algorithms are designed to address single tasks, the development of algorithms that facilitate transfer learning is a topic of ongoing interest in the machine-learning community. This chapter provides an introduction to the goals, formulations, and challenges of transfer learning. It surveys current research in this area, giving an overview of the state of the art and outlining the open problems. The survey covers transfer in both inductive learning and reinforcement learning, and discusses the issues of negative transfer and task mapping in depth.

INTRODUCTION

Human learners appear to have inherent ways to transfer knowledge between tasks. That is, we recognize and apply relevant knowledge from previous learning experiences when we encounter new tasks. The more related a new task is to our previous experience, the more easily we can master it.

Common machine learning algorithms, in contrast, traditionally address isolated tasks. *Transfer learning* attempts to change this by developing methods to transfer knowledge learned in one or more *source tasks* and use it to improve learning in a related *target task* (see Figure 1). Techniques that enable knowledge transfer represent progress towards making machine learning as efficient as human learning.

This chapter provides an introduction to the goals, formulations, and challenges of transfer learning. It surveys current research in this area, giving an overview of the state of the art and outlining the open problems.

Transfer methods tend to be highly dependent on the machine learning algorithms being used to learn the tasks, and can often simply be considered extensions of those algorithms. Some work in transfer learning is in the context of inductive learning, and involves extending well-known classification and inference algorithms such as neural networks, Bayesian networks, and Markov Logic Networks. Another major area is in the context of reinforcement learning, and involves extending algorithms such as Q-learning and policy search. This chapter surveys these areas separately.

Appears in the *Handbook of Research on Machine Learning Applications*, published by IGI Global, edited by E. Soria, J. Martin, R. Magdalena, M. Martinez and A. Serrano, 2009.

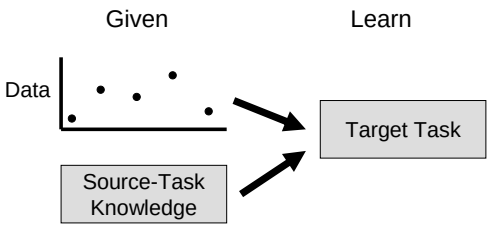

Fig. 1. Transfer learning is machine learning with an additional source of information apart from the standard training data: knowledge from one or more related tasks.

The goal of transfer learning is to improve learning in the target task by leveraging knowledge from the source task. There are three common measures by which transfer might improve learning. First is the initial performance achievable in the target task using only the transferred knowledge, before any further learning is done, compared to the initial performance of an ignorant agent. Second is the amount of time it takes to fully learn the target task given the transferred knowledge compared to the amount of time to learn it from scratch. Third is the final performance level achievable in the target task compared to the final level without transfer. Figure 2 illustrates these three measures.

If a transfer method actually decreases performance, then *negative transfer* has occurred. One of the major challenges in developing transfer methods is to produce positive transfer between appropriately related tasks while avoiding negative transfer between tasks that are less related. A section of this chapter discusses approaches for avoiding negative transfer.

When an agent applies knowledge from one task in another, it is often necessary to map the characteristics of one task onto those of the other to specify correspondences. In much of the work on transfer learning, a human provides this *mapping*, but some methods provide ways to perform the mapping automatically. Another section of the chapter discusses work in this area.

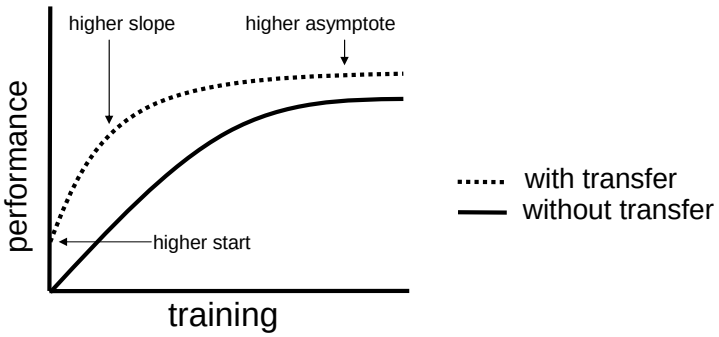

Fig. 2. Three ways in which transfer might improve learning.

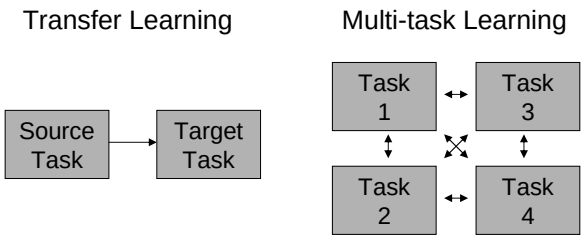

Fig. 3. As we define transfer learning, the information flows in one direction only, from the source task to the target task. In multi-task learning, information can flow freely among all tasks.

We will make a distinction between transfer learning and *multi-task learning* [5], in which several tasks are learned simultaneously (see Figure 3). Multi-task learning is clearly closely related to transfer, but it does not involve designated source and target tasks; instead the learning agent receives information about several tasks at once. In contrast, by our definition of transfer learning, the agent knows nothing about a target task (or even that there will be a target task) when it learns a source task. It may be possible to approach a multi-task learning problem with a transfer-learning method, but the reverse is not possible. It is useful to make this distinction because a learning agent in a real-world setting is more likely to encounter transfer scenarios than multi-task scenarios.

TRANSFER IN INDUCTIVE LEARNING

In an inductive learning task, the objective is to induce a predictive model from a set of training examples [28]. Often the goal is classification, i.e. assigning class labels to examples. Examples of classification systems are artificial neural networks and symbolic rule-learners. Another type of inductive learning involves modeling probability distributions over interrelated variables, usually with graphical models. Examples of these systems are Bayesian networks and Markov Logic Networks [34].

The predictive model learned by an inductive learning algorithm should make accurate predictions not just on the training examples, but also on future examples that come from the same distribution. In order to produce a model with this generalization capability, a learning algorithm must have an *inductive bias* [28] – a set of assumptions about the true distribution of the training data.

The bias of an algorithm is often based on the *hypothesis space* of possible models that it considers. For example, the hypothesis space of the Naive Bayes model is limited by the assumption that example characteristics are conditionally independent given the class of an example. The bias of an algorithm can also be determined by its search process through the hypothesis space, which determines the order in which hypotheses are considered. For example, rule-learning algorithms typically construct rules one predicate at a time, which reflects the

assumption that predicates contribute significantly to example coverage by themselves rather than in pairs or more.

Transfer in inductive learning works by allowing source-task knowledge to affect the target task's inductive bias. It is usually concerned with improving the speed with which a model is learned, or with improving its generalization capability. The next subsection discusses inductive transfer, and the following ones elaborate on three specific settings for inductive transfer.

There is some related work that is not discussed here because it specifically addresses multi-task learning. For example, Niculescu-Mizil and Caruana [29] learn Bayesian networks simultaneously for multiple related tasks by biasing learning toward similar structures for each task. While this is clearly related to transfer learning, it is not directly applicable to the scenario in which a target task is encountered after one or more source tasks have already been learned.

Inductive Transfer

In *inductive transfer* methods, the target-task inductive bias is chosen or adjusted based on the source-task knowledge (see Figure 4). The way this is done varies depending on which inductive learning algorithm is used to learn the source and target tasks. Some transfer methods narrow the hypothesis space, limiting the possible models, or remove search steps from consideration. Other methods broaden the space, allowing the search to discover more complex models, or add new search steps.

Baxter [2] frames the transfer problem as that of choosing one hypothesis space from a family of spaces. By solving a set of related source tasks in each hypothesis space of the family and determining which one produces the best overall generalization error, he selects the most promising space in the family for a target task. Baxter's work, unlike most in transfer learning, includes theoretical as well as experimental results. He derives bounds on the number of source tasks and examples needed to learn an inductive bias, and on the generalization capability of a target-task solution given the number of source tasks and examples in each task.

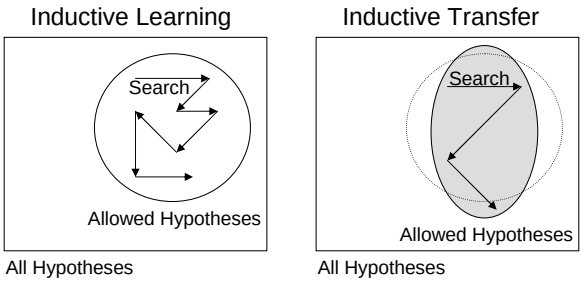

Fig. 4. Inductive learning can be viewed as a directed search through a specified hypothesis space [28]. Inductive transfer uses source-task knowledge to adjust the inductive bias, which could involve changing the hypothesis space or the search steps.

Thrun and Mitchell [55] look at solving Boolean classification tasks in a lifelong-learning framework, where an agent encounters a collection of related problems over its lifetime. They learn each new task with a neural network, but they enhance the standard gradient-descent algorithm with slope information acquired from previous tasks. This speeds up the search for network parameters in a target task and biases it towards the parameters for previous tasks.

Mihalkova and Mooney [27] perform transfer between Markov Logic Networks. Given a learned MLN for a source task, they learn an MLN for a related target task by starting with the source-task one and diagnosing each formula, adjusting ones that are too general or too specific in the target domain. The hypothesis space for the target task is therefore defined in relation to the source-task MLN by the operators that generalize or specify formulas.

Hlynsson [17] phrases transfer learning in classification as a minimum description length problem given source-task hypotheses and target-task data. That is, the chosen hypothesis for a new task can use hypotheses for old tasks but stipulate exceptions for some data points in the new task. This method aims for a tradeoff between accuracy and compactness in the new hypothesis.

Ben-David and Schuller [3] propose a transformation framework to determine how related two Boolean classification tasks are. They define two tasks as related with respect to a class of transformations if they are equivalent under that class; that is, if a series of transformations can make one task look exactly like the other. They provide conditions under which learning related tasks concurrently requires fewer examples than single-task learning.

Bayesian Transfer

One area of inductive transfer applies specifically to Bayesian learning methods. Bayesian learning involves modeling probability distributions and taking advantage of conditional independence among variables to simplify the model. An additional aspect that Bayesian models often have is a *prior distribution*, which describes the assumptions one can make about a domain before seeing any training data. Given the data, a Bayesian model makes predictions by combining it with the prior distribution to produce a *posterior distribution*. A strong prior can significantly affect these results (see Figure 5). This serves as a natural way for Bayesian learning methods to incorporate prior knowledge – in the case of transfer learning, source-task knowledge.

Marx et al. [24] use a Bayesian transfer method for tasks solved by a logistic regression classifier. The usual prior for this classifier is a Gaussian distribution with a mean and variance set through cross-validation. To perform transfer, they instead estimate the mean and variance by averaging over several source tasks. Raina et al. [33] use a similar approach for multi-class classification by learning a multivariate Gaussian prior from several source tasks.

Dai et al. [7] apply a Bayesian transfer method to a Naive Bayes classifier. They set the initial probability parameters based on a single source task, and revise them using target-task data. They also provide some theoretical bounds on the prediction error and convergence rate of their algorithm.

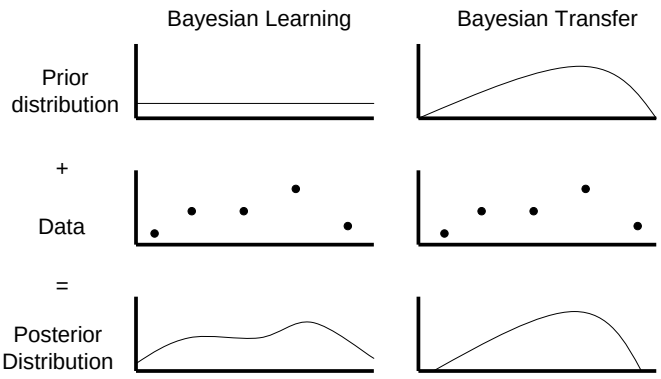

Fig. 5. Bayesian learning uses a prior distribution to smooth the estimates from training data. Bayesian transfer may provide a more informative prior from source-task knowledge.

Hierarchical Transfer

Another setting for transfer in inductive learning is *hierarchical transfer*. In this setting, solutions to simple tasks are combined or provided as tools to produce a solution to a more complex task (see Figure 6). This can involve many tasks of varying complexity, rather than just a single source and target. The target task might use entire source-task solutions as parts of its own, or it might use them in a more subtle way to improve learning.

Sutton and McCallum [43] begin with a sequential approach where the prediction for each task is used as a feature when learning the next task. They then proceed to turn the problem into a multi-task learning problem by combining all the models and applying them jointly, which brings their method outside our definition of transfer learning, but the initial sequential approach is an example of hierarchical transfer.

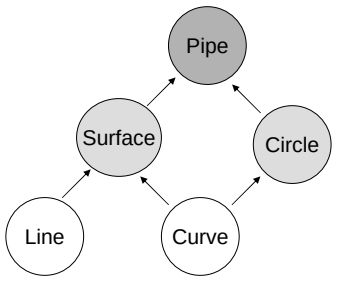

Fig. 6. An example of a concept hierarchy that could be used for hierarchical transfer, in which solutions from simple tasks are used to help learn a solution to a more complex task. Here the simple tasks involve recognizing lines and curves in images, and the more complex tasks involve recognizing surfaces, circles, and finally pipe shapes.

Stracuzzi [42] looks at the problem of choosing relevant source-task Boolean concepts from a knowledge base to use while learning more complex concepts. He learns rules to express concepts from a stream of examples, allowing existing concepts to be used if they help to classify the examples, and adds and removes dependencies between concepts in the knowledge base.

Taylor et al. [49] propose a transfer hierarchy that orders tasks by difficulty, so that an agent can learn them in sequence via inductive transfer. By putting tasks in order of increasing difficulty, they aim to make transfer more effective. This approach may be more applicable to the multi-task learning scenario, since by our definition of transfer learning the agent may not be able to choose the order in which it learns tasks, but it could be applied to help choose from an existing set of source tasks.

Transfer with Missing Data or Class Labels

Inductive transfer can be viewed not only as a way to improve learning in a standard supervised-learning task, but also as a way to offset the difficulties posed by tasks that involve unsupervised learning, semi-supervised learning, or small datasets. That is, if there are small amounts of data or class labels for a task, treating it as a target task and performing inductive transfer from a related source task can lead to more accurate models. These approaches therefore use source-task data to enhance target-task data, despite the fact that the two datasets are assumed to come from different probability distributions.

The Bayesian transfer methods of Dai et al. [7] and Raina et al. [33] are intended to compensate for small amounts of target-task data. One of the benefits of Bayesian learning is the stability that a prior distribution can provide in the absence of large datasets. By estimating a prior from related source tasks, these approaches prevent the overfitting that would tend to occur with limited data.

Dai et al. [8] address transfer learning in a boosting algorithm using large amounts of data from a previous task to supplement small amounts of new data. Boosting is a technique for learning several weak classifiers and combining them to form a stronger classifier [16]. After each classifier is learned, the examples are reweighted so that later classifiers focus more on examples the previous ones misclassified. Dai et al. extend this principle by also weighting source-task examples according to their similarity to target-task examples. This allows the algorithm to leverage source-task data that is applicable to the target task while paying less attention to data that appears less useful.

Shi et al. [39] look at transfer learning in unsupervised and semi-supervised settings. They assume that a reasonably sized dataset exists in the target task, but it is largely unlabeled due to the expense of having an expert assign labels. To address this problem they propose an active learning approach, in which the target-task learner requests labels for examples only when necessary. They construct a classifier with labeled examples, including mostly source-task ones, and estimate the confidence with which this classifer can label the unknown examples. When the confidence is too low, they request an expert label.

TRANSFER IN REINFORCEMENT LEARNING

A reinforcement learning (RL) agent operates in a sequential-control environment called a Markov decision process (MDP) [45]. It senses the *state* of the environment and performs *actions* that change the state and also trigger *rewards*. Its objective is to learn a *policy* for acting in order to maximize its cumulative reward. This involves solving a temporal credit-assignment problem, since an entire sequence of actions may be responsible for a single immediate reward.

A typical RL agent behaves according to the diagram in Figure 7. At time step t, it observes the current state s_t and consults its current policy π to choose an action, $\pi(s_t) = a_t$. After taking the action, it receives a reward r_t and observes the new state s_{t+1}, and it uses that information to update its policy before repeating the cycle. Often RL consists of a sequence of *episodes*, which end whenever the agent reaches one of a set of ending states.

During learning, the agent must balance between *exploiting* the current policy (acting in areas that it knows to have high rewards) and *exploring* new areas to find potentially higher rewards. A common solution is the ϵ-greedy method, in which the agent takes random exploratory actions a small fraction of the time ($\epsilon << 1$), but usually takes the action recommended by the current policy.

There are several categories of RL algorithms. Some types of methods are only applicable when the agent knows its environment model (the reward function and the state transition function). In this case dynamic programming can solve directly for the optimal policy without requiring any interaction with the environment. In most RL problems, however, the model is unknown. *Model-learning* approaches use interaction with the environment to build an approximation of the true model. *Model-free* approaches learn to act without ever explicitly modeling the environment.

Temporal-difference methods [44] operate by maintaining and iteratively updating *value functions* to predict the rewards earned by actions. They begin with an inaccurate function and update it based on interaction with the environment, propagating reward information back along action sequences. One popular method is Q-learning [62], which involves learning a function $Q(s, a)$ that estimates the cumulative reward starting in state s and taking action a and following the current policy thereafter. Given the optimal Q-function, the

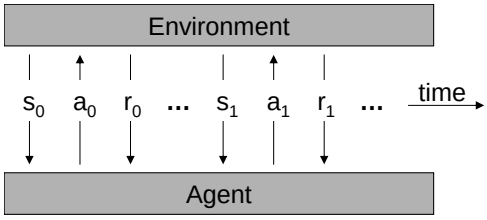

Fig. 7. A reinforcement learning agent interacts with its environment: it receives information about its state (s), chooses an action to take (a), receives a reward (r), and then repeats.

optimal policy is to take the action corresponding to $argmax_a Q(s_t, a)$. When there are small finite numbers of states and actions, the Q-function can be represented explicitly as a table. In domains that have large or infinite state spaces, a function approximator such as a neural network or support-vector machine can be used to represent the Q-function.

Policy-search methods, instead of maintaining a function upon which a policy is based, maintain and update a policy directly. They begin with an inaccurate policy and update it based on interaction with the environment. Heuristic search and optimization through gradient descent are among the approaches that can be used in policy search.

Transfer in RL is concerned with speeding up the learning process, since RL agents can spend many episodes doing random exploration before acquiring a reasonable Q-function. We divide RL transfer into five categories that represent progressively larger changes to existing RL algorithms. The subsections below describe those categories and present examples from published research.

Starting-Point Methods

Since all RL methods begin with an initial solution and then update it through experience, one straightforward type of transfer in RL is to set the initial solution in a target task based on knowledge from a source task (see Figure 8). Compared to the random or zero setting that RL algorithms usually use at first, these *starting-point methods* can begin the RL process at a point much closer to a good target-task solution. There are variations on how to use the source-task knowledge to set the initial solution, but in general the RL algorithm in the target task is unchanged.

Taylor et al. [53] use a starting-point method for transfer in temporal-difference RL. To perform transfer, they copy the final value function of the source task and use it as the initial one for the target task. As many transfer approaches do, this requires a mapping of features and actions between the tasks, and they provide a mapping based on their domain knowledge.

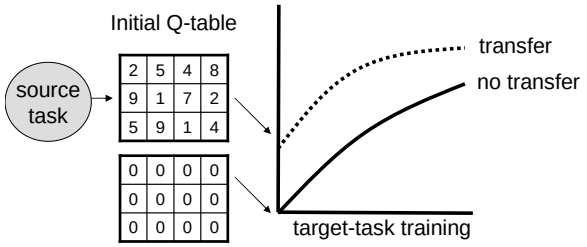

Fig. 8. Starting-point methods for RL transfer set the initial solution based on the source task in the hope of starting at a higher performance level than the typical initial solution would. In this example, a Q-function table is initialized to a source-task table, and the target-task performance begins at a level that is only reached after some training when beginning with a typical all-zero table.

Tanaka and Yamamura [47] use a similar approach in temporal-difference learning without function approximation, where value functions are simply represented by tables. This greater simplicity allows them to combine knowledge from several source tasks: they initialize the value table of the target task to the average of tables from several prior tasks. Furthermore, they use the standard deviations from prior tasks to determine priorities between temporal-difference backups.

Approaching temporal-difference RL as a batch problem instead of an incremental one allows for different kinds of starting-point transfer methods. In batch RL, the agent interacts with the environment for more than one step or episode at a time before updating its solution. Lazaric et al. [21] perform transfer in this setting by finding source-task samples that are similar to the target task and adding them to the normal target-task samples in each batch, thus increasing the available data early on. The early solutions are almost entirely based on source-task knowledge, but the impact decreases in later batches as more target-task data becomes available.

Moving away from temporal-difference RL, starting-point methods can take even more forms. In a model-learning Bayesian RL algorithm, Wilson et al. [63] perform transfer by treating the distribution of previous MDPs as a prior for the current MDP. In a policy-search genetic algorithm, Taylor et al. [54] transfer a population of policies from a source task to serve as the initial population for a target task.

Imitation Methods

Another class of RL transfer methods involves applying the source-task policy to choose some actions while learning the target task (see Figure 9). While they make no direct changes to the target-task solution the way that starting-point methods do, these *imitation methods* affect the developing solution by producing different function or policy updates. Compared to the random exploration that RL algorithms typically do, decisions based on a source-task policy can lead the agent more quickly to promising areas of the environment.

One method is to follow a source-task policy only during exploration steps of the target task, when the agent would otherwise be taking a random action. Madden and Howley [23] use this approach in tabular Q-learning. They represent a source-task policy as a set of rules in propositional logic and choose actions based on those rules during exploration steps.

Fernandez and Veloso [15] instead give the agent a three-way choice between exploiting the current target-task policy, exploiting a past policy, and exploring randomly. They introduce a second parameter, in addition to the ϵ of ϵ-greedy exploration, to determine the probability of making each choice.

Another imitation method called *demonstration* involves following a source-task policy for a fixed number of episodes at the beginning of the target task and then reverting to normal RL. In the early steps of the target task, the current policy can be so ill-formed that exploiting it is no different than exploring randomly. This approach aims to avoid that initial uncertainty and to generate

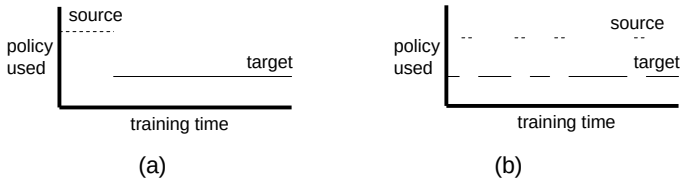

Fig. 9. Imitation methods for RL transfer follow the source-task policy during some steps of the target task. The imitation steps may all occur at the beginning of the target task, as in (a) above, or they may be interspersed with steps that follow the developing target-task policy, as in (b) above.

enough data to create a reasonable target-task policy by the time the demonstration period ends. Torrey et al. [58] and Torrey et al. [56] perform transfer via demonstration, representing the source-task policy as a relational finite-state machine and a Markov Logic Network respectively.

Hierarchical Methods

A third class of RL transfer includes *hierarchical methods*. These view the source as a subtask of the target, and use the solution to the source as a building block for learning the target. Methods in this class have strong connections to the area of hierarchical RL, in which a complex task is learned in pieces through division into a hierarchy of subtasks (see Figure 10).

An early approach of this type is to compose several source-task solutions to form a target-task solution, as is done by Singh [40]. He addresses a scenario in which complex tasks are temporal concatenations of simple ones, so that a target task can be solved by a composition of several smaller solutions.

Mehta et al. [25] have a transfer method that works directly within the hierarchical RL framework. They learn a task hierarchy by observing successful behavior in a source task, and then use it to apply the MaxQ hierarchical RL algorithm [10] in the target task. This removes the burden of designing a task hierarchy through transfer.

Other approaches operate within the framework of *options*, which is a term for temporally-extended actions in RL [31]. An option typically consists of a starting condition, an ending condition, and an internal policy for choosing lower-level actions. An RL agent treats each option as an additional action along with the original lower-level ones (see Figure 10).

In some scenarios it may be useful to have the entire source-task policy as an option in the target task, as Croonenborghs et al. [6] do. They learn a relational decision tree to represent the source-task policy and allow the target-task learner to execute it as an option. Another possibility is to learn smaller options, either during or after the process of learning the source task, and offer them to the target. Asadi and Huber [1] do this by finding frequently-visited states in the source task to serve as ending conditions for options.

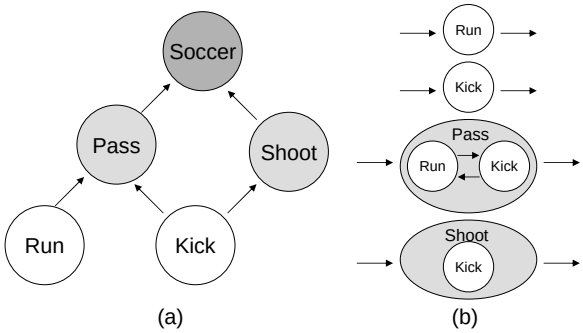

Fig. 10. (a) An example of a task hierarchy that could be used to train agents to play soccer via hierarchical RL. Lower-level abilities like kicking a ball and running are needed for higher-level abilities like passing and shooting, which could then be combined to learn to play soccer. (b) The mid-level abilities represented as options alongside the low-level actions.

Alteration Methods

The next class of RL transfer methods involves altering the state space, action space, or reward function of the target task based on source-task knowledge. These *alteration methods* have some overlap with option-based transfer, which also changes the action space in the target task, but they include a wide range of other approaches as well.

One way to alter the target-task state space is to simplify it through state abstraction. Walsh et al. [60] do this by aggregating over comparable source-task states. They then use the aggregate states to learn the target task, which reduces the complexity significantly.

There are also approaches that expand the target-task state space instead of reducing it. Taylor and Stone [51] do this by adding a new state variable in the target task. They learn a decision list that represents the source-task policy and use its output as the new state variable.

While option-based transfer methods add to the target-task action space, there is also some work in decreasing the action space. Sherstov and Stone [38] do this by evaluating in the source task which of a large set of actions are most useful. They then consider only a smaller action set in the target task, which decreases the complexity of the value function significantly and also decreases the amount of exploration needed.

Reward shaping is a design technique in RL that aims to speed up learning by providing immediate rewards that are more indicative of cumulative rewards. Usually it requires human effort, as many aspects of RL task design do. Konidaris and Barto [19] do reward shaping automatically through transfer. They learn to predict rewards in the source task and use this information to create a shaped reward function in the target task.

New RL Algorithms

A final class of RL transfer methods consists of entirely new RL algorithms. Rather than making small additions to an existing algorithm or making changes to the target task, these approaches address transfer as an inherent part of RL. They incorporate prior knowledge as an intrinsic part of the algorithm.

Price and Boutilier [32] propose a temporal-difference algorithm in which value functions are influenced by observations of expert agents. They use a variant of the usual value-function update calculation that includes an expert's experience, weighted by the agent's confidence in itself and in the expert. They also perform extra backups at states the expert visits to focus attention on those areas of the state space.

There are several algorithms for case-based RL that accomodate transfer. Sharma et al. [37] propose one in which Q-functions are estimated using a Gaussian kernel over stored cases in a library. Cases are added to the library from both the source and target tasks when their distance to their nearest neighbor is above a threshold. Taylor et al. [48] use source-task examples more selectively in their case-based RL algorithm. They use target-task cases to make decisions when there are enough, and only use source-task examples when insufficient target examples exist.

Torrey et al. [57, 59] use an RL algorithm called Knowledge-Based Kernel Regression (KBKR) that allows transfer via advice-taking. *Advice* in this algorithm is a rule telling the agent which action to prefer in a set of states described by a conjunct of predicates. KBKR approximates the Q-function with a support-vector machine and includes advice as a soft constraint. The Q-function, which is relearned in batches using temporal-difference updates, trades off between matching the agent's experience and matching the advice. Torrey et al. generate advice through automated analysis of the source-task solution; in [59] they construct advice directly from the source-task Q-function, and in [57] they learn rules in first-order logic by observing the source-task agent as it follows its policy.

AVOIDING NEGATIVE TRANSFER

Given a target task, the effectiveness of any transfer method depends on the source task and how it is related to the target. If the relationship is strong and the transfer method can take advantage of it, the performance in the target task can significantly improve through transfer. However, if the source task is not sufficiently related or if the relationship is not well leveraged by the transfer method, the performance with many approaches may not only fail to improve – it may actually decrease. This section examines work on preventing transfer from negatively affecting performance.

Ideally, a transfer method would produce positive transfer between appropriately related tasks while avoiding negative transfer when the tasks are not a good match. In practice, these goals are difficult to achieve simultaneously. Approaches that have safeguards to avoid negative transfer often produce a smaller

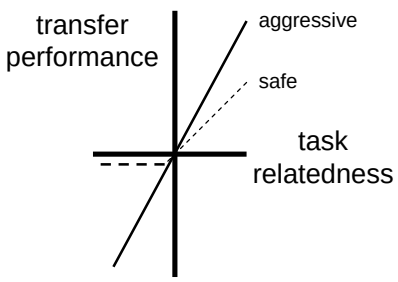

Fig. 11. A representation of how the degree of relatedness between the source and target tasks translates to target-task performance when conducting transfer from the source task. With aggressive approaches, there can be higher benefits at high degrees of relatedness, but there can also be negative transfer at low levels. Safer approaches may limit negative transfer at the lower end, but may also have fewer benefits at the higher end.

effect from positive transfer due to their caution. Conversely, approaches that transfer aggressively and produce large positive-transfer effects often have no protection against negative transfer (see Figure 11).

For example, consider the imitation methods for RL transfer. On one end of the range an agent imitates a source-task policy only during infrequent exploration steps, and on the other end it demonstrates the source-task policy for a fixed number of initial episodes. The exploration method is very cautious and therefore unlikely to produce negative transfer, but it is also unlikely to produce large initial performance increases. The demonstration method is very aggressive; if the source-task policy is a poor one for the target task, following it blindly will produce negative transfer. However, when the source-task solution is a decent one for the target task, it can produce some of the largest initial performance improvements of any method.

Rejecting Bad Information

One way of approaching negative transfer is to attempt to recognize and reject harmful source-task knowledge while learning the target task. The goal in this approach is to minimize the impact of bad information, so that the transfer performance is at least no worse than learning the target task without transfer. At the extreme end, an agent might disregard the transferred knowledge completely, but some methods also allow it to selectively reject parts and keep other parts.

Option-based transfer in reinforcement learning (e.g. Croonenborghs et al. [6]) is an example of an approach that naturally incorporates the ability to reject bad information. Since options are treated as additional actions, the agent can choose to use them or not to use them; in Q-learning, for example, agents learn Q-values for options just as for native actions. If an option regularly produces poor performance, its Q-values will degrade and the agent will choose it less frequently. However, if an option regularly leads to good results, its Q-values will

grow and the agent will choose it more often. Option-based transfer can therefore provide a good balance between achieving positive transfer and avoiding negative transfer.

A specific approach that incorporates the ability to reject bad information is the KBKR advice-taking algorithm for transfer in reinforcement learning [57, 59]. Recall that KBKR approximates the Q-function with a support-vector machine and includes advice from the source task as a soft constraint. Since the Q-function trades off between matching the agent's experience and matching the advice, the agent can learn to disregard advice that disagrees with its experience.

Rosenstein et al. [35] present an approach for detecting negative transfer in naive Bayes classification tasks. They learn a hyperprior for both the source and target tasks, and the variance of this hyperprior is proportional to the dissimilarity between the tasks. It may be possible to use a method like this to decide whether to transfer at all, by setting an acceptable threshold of similarity.

Choosing a Source Task

There are more possibilities for avoiding negative transfer if there exists not just one source task, but a set of candidate source tasks. In this case the problem becomes choosing the best source task (see Figure 12). Transfer methods without much protection against negative transfer may still be effective in this scenario, as long as the best source task is at least a decent match.

An example of this approach is the previously-mentioned transfer hierarchy of Taylor et al. [49], who order tasks by difficulty. Appropriate source tasks are usually less difficult than the target task, but not so much simpler that they contain little information. Given a task ordering, it may be possible to locate the position of the target task in the hierarchy and select a source task that is only moderately less difficult.

Talvitie and Singh [46] use a straightforward method of selecting a previous Markov decision process to transfer. They run each candidate MDP in the target task for a fixed length of time and order them by their performance. Then they select the best one and continue with it, only proceeding down the list if the current MDP begins performing significantly worse than it originally appeared. This trial-and-error approach, though it may be costly in the aggregate number of training episodes needed, is simple and widely applicable.

Kuhlmann and Stone [20] look at finding similar tasks when each task is specified in a formal language. They construct a graph to represent the elements and rules of a task. This allows them to find identical tasks by checking for graph isomorphism, and by creating minor variants of a target-task graph, they can also search for similar tasks. If they find an isomorphic match, they conduct value-function transfer.

Eaton and DesJardins [12] propose choosing from among candidate solutions to a source task rather than from among candidate source tasks. Their setting is multi-resolution learning, where a classification task is solved by an ensemble of models that vary in complexity. Low-resolution models are simple and coarse, while higher-resolution models are more complex and detailed. They reason that

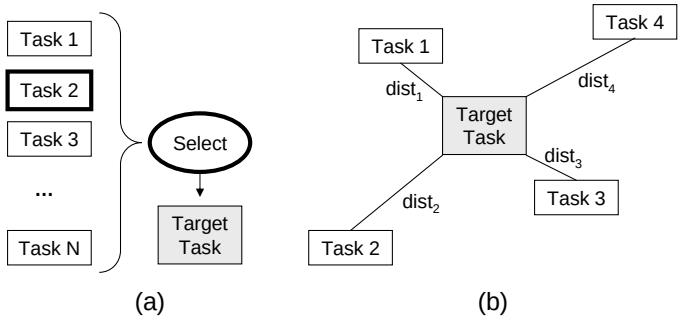

Fig. 12. (a) One way to avoid negative transfer is to choose a good source task from which to transfer. In this example, Task 2 is selected as being the most related. (b) Another way to avoid negative transfer is to model the way source tasks are related to the target task and combine knowledge from them with those relationships in mind.

high-resolution models are less transferrable between tasks, and select a resolution below which to share models with a target task.

Modeling Task Similarity

Given multiple candidate source tasks, it may be beneficial to use several or all of them rather than to choose just one (see Figure 12). Some approaches discussed in this chapter do this naively, without evaluating how the source tasks are related to the target. However, there are some approaches that explicitly model relationships between tasks and include this information in the transfer method. This can lead to better use of source-task knowledge and decrease the risk of negative transfer.

Carroll and Seppi [4] develop several similarity measures for reinforcement learning tasks, comparing policies, value functions, and rewards. These are only measurable while the target task is being learned, so their practical use in transfer scenarios is limited. However, they make the relevant point that task similarity is intimately linked with a particular transfer method, and cannot be evaluated independently.

Eaton et al. [13] construct a graph in which nodes represent source tasks and distances represent a transferability metric. Given a new inductive learning task, they estimate parameters by fitting the task into the graph and learning a function that translates graph locations to task parameters. This method not only models the relationships between tasks explicitly, but also gives an algorithm for the informed use of several source tasks in transfer learning.

Ruckert and Kramer [36] look at inductive transfer via kernel methods. They learn a meta-kernel that serves as a similarity function between tasks. Given this and a set of kernels that perform well in source tasks, they perform numerical optimization to construct a kernel for a target task. This approach determines the inductive bias in the target task (the kernel) by combining information from several source tasks whose relationships to the target are known.

AUTOMATICALLY MAPPING TASKS

An inherent aspect of transfer learning is recognizing the correspondences between tasks. Knowledge from one task can only be applied to another if it is expressed in a way that the target-task agent understands. In some cases, the representations of the tasks are assumed to be identical, or at least one is a subset of the other. Otherwise, a *mapping* is needed to translate between task representations (see Figure 13).

Many transfer approaches do not address the mapping problem directly and require that a human provide this information. However, there are some transfer approaches that do address the mapping problem. This section discusses some of this work.

Equalizing Task Representations

For some transfer scenarios, it may be possible to avoid the mapping problem altogether by ensuring that the source and target tasks have the same representation. If the language of the source-task knowledge is identical to or a subset of the language of the target task, it can be applied directly with no translation. Sometimes a domain can be constructed so that this occurs naturally, or a common representation that equalizes the tasks can be found.

Relational learning is useful for creating domains that naturally produce common task representations. First-order logic represents objects in a domain with symbolic variables, which can allow abstraction that the more typical propositional feature vector cannot. Driessens et al. [11] show how relational reinforcement learning can simplify transfer in RL.

Another framework for constructing a domain relationally is that of Konidaris and Barto [19], who express knowledge in two different spaces. In *agent space* the representation is constant across tasks, while in *problem space* it is task-dependent. They transfer agent-space knowledge only because its common representation makes it straightforward to transfer.

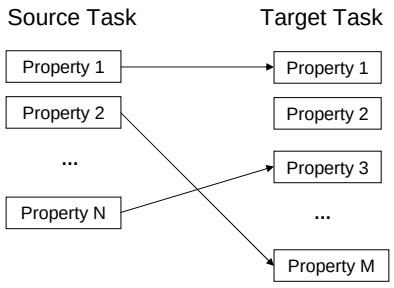

Fig. 13. A mapping generally translates source-task properties into target-task properties. The numbers of properties may not be equal in the two tasks, and the mapping may not be one-to-one. Properties may include entries in a feature vector, objects in a relational world, RL actions, etc.

Pan et al. [30] take a mathematical approach to finding a common representation for two separate classification tasks. They use kernel methods to find a low-dimensional feature space where the distributions of source and target data are similar, and transfer a source-task model for this smaller space. This approach stretches our strict definition of transfer learning, which assumes the target task is unknown when the source task is learned, but in some scenarios it may be practical to adjust the source-task solution to a different feature space after gaining some knowledge about the target task.

Trying Multiple Mappings

One straightforward way of solving the mapping problem is to generate several possible mappings and allow the target-task agent to try them all. The candidate mappings can be an exhaustive set, or they can be limited by constraints on what elements are permissible matches for other elements. Exhaustive sets may be computationally infeasible for large domains.

Taylor et al. [50] perform an exhaustive search of possible mappings in RL transfer. They evaluate each candidate using a small number of episodes in the target task, and select the best one to continue learning. Mihalkova et al. [26] limit their search for mappings in MLN transfer, requiring that mapped predicates have matching arity and argument types. Under those constraints, they conduct an exhaustive search to find the best mapping between networks.

Soni and Singh [41] not only limit the candidate mappings by considering object types, but also avoid a separate evaluation of each mapping by using options in RL transfer. They generate a set of possible mappings by connecting target-task objects to all source-task objects of the matching type. With each mapping, they create an option from a source MDP. The options framework gives an inherent way to compare multiple mappings while learning a target MDP without requiring extra trial periods.

Mapping by Analogy

If the task representations must differ, and the scenario calls for choosing one mapping rather than trying multiple candidates, then there are some methods that construct a mapping by analogy. These methods examine the characteristics of the source and target tasks and find elements that correspond. For example, in reinforcement learning, actions that correspond produce similar rewards and state changes, and objects that correspond are affected similarly by actions.

Analogical structure mapping [14] is a generic procedure based on cognitive theories of analogy that finds corresponding elements. It assigns scores to local matches and searches for a global match that maximizes the scores; permissible matches and scoring functions are domain-dependent. Several transfer approaches use this framework solve the mapping problem. Klenk and Forbus [18] apply it to solve physics problems that are written in a predicate-calculus language by retrieving and forming analogies from worked solutions written in the

same language. Liu and Stone [22] apply it in reinforcement learning to find matching features and actions between tasks.

There are also some approaches that rely more on statistical analysis than on logical reasoning to find matching elements. Taylor and Stone [52] learn mappings for RL tasks by running a small number of target-task episodes and then training classifiers to characterize actions and objects. If a classifier trained for one action predicts the results of another action well, then those actions are mapped; likewise, if a classifier trained for one object predicts the behavior of another object well, those objects are mapped. Wang and Mahadevan [61] translate datasets to low-dimensional feature spaces using dimensionality reduction, and then perform a statistical shaping technique called Procrustes analysis to align the feature spaces.

THE FUTURE OF TRANSFER LEARNING

The challenges discussed in this chapter will remain relevant in future work on transfer learning, particularly the avoidance of negative transfer and the automation of task mapping. Humans appear to have mechanisms for deciding when to transfer information, selecting appropriate sources of knowledge, and determining the appropriate level of abstraction. It is not always clear how to make these decisions for a single machine learning algorithm, much less in general.

Another challenge for future work is to enable transfer between more diverse tasks. Davis and Domingos [9] provide a potential direction for this in their work on MLN transfer. They perform pattern discovery in the source task to find second-order formulas, which represent universal abstract concepts like symmetry and transitivity. When learning an MLN for the target task, they allow the search to use the discovered formulas in addition to the original predicates in the domain. This approach is recognizeable as inductive transfer, but the source-task knowledge is highly abstract, which allows the source and target tasks to differ significantly.

Yet another challenge is to perform transfer in more complex testbeds. Particularly in RL transfer, it can become much more difficult to achieve transfer as the source and target tasks become more complex. Since practical applications of reinforcement learning are likely to be highly complex, it is important not to limit research on RL tranfer to simple domains.

Transfer learning has become a sizeable subfield in machine learning. It has ideological benefits, because it is seen as an important aspect of human learning, and also practical benefits, because it can make machine learning more efficient. As computing power increases and researchers apply machine learning to more complex problems, knowledge transfer can only become more desirable.

ACKNOWLEDGEMENTS

This chapter was written while the authors were partially supported by DARPA grants HR0011-07-C-0060 and FA8650-06-C-7606.

References

1. M. Asadi and M. Huber. Effective control knowledge transfer through learning skill and representation hierarchies. In *International Joint Conference on Artificial Intelligence*, 2007.
2. J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.
3. S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Conference on Learning Theory*, 2003.
4. C. Carroll and K. Seppi. Task similarity measures for transfer in reinforcement learning task libraries. In *IEEE International Joint Conference on Neural Networks*, 2005.
5. R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.
6. T. Croonenborghs, K. Driessens, and M. Bruynooghe. Learning relational skills for inductive transfer in relational reinforcement learning. In *International Conference on Inductive Logic Programming*, 2007.
7. W. Dai, G. Xue, Q. Yang, and Y. Yu. Transferring Naive Bayes classifiers for text classification. In *AAAI Conference on Artificial Intelligence*, 2007.
8. W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. In *International Conference on Machine Learning*, 2007.
9. J. Davis and P. Domingos. Deep transfer via second-order Markov logic. In *AAAI Workshop on Transfer Learning for Complex Tasks*, 2008.
10. T. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.
11. K. Driessens, J. Ramon, and T. Croonenborghs. Transfer learning for reinforcement learning through goal and policy parametrization. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.
12. E. Eaton and M. DesJardins. Knowledge transfer with a multiresolution ensemble of classifiers. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.
13. E. Eaton, M. DesJardins, and T. Lane. Modeling transfer relationships between learning tasks for improved inductive transfer. In *European Conference on Machine Learning*, 2008.
14. B. Falkenhainer, K. Forbus, and D. Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.
15. F. Fernandez and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Conference on Autonomous Agents and Multi-Agent Systems*, 2006.
16. Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
17. H. Hlynsson. Transfer learning using the minimum description length principle with a decision tree application. Master's thesis, University of Amsterdam, 2007.
18. M. Klenk and K. Forbus. Measuring the level of transfer learning by an AP physics problem-solver. In *AAAI Conference on Artificial Intelligence*, 2007.
19. G. Konidaris and A. Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *International Conference on Machine Learning*, 2006.
20. G. Kuhlmann and P. Stone. Graph-based domain mapping for transfer learning in general games. In *European Conference on Machine Learning*, 2007.
21. A. Lazaric, M. Restelli, and A. Bonarini. Transfer of samples in batch reinforcement learning. In *International Conference on Machine Learning*, 2008.

22. Y. Liu and P. Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *AAAI Conference on Artificial Intelligence*, 2006.
23. M. Madden and T. Howley. Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, 21:375–398, 2004.
24. Z. Marx, M. Rosenstein, L. Kaelbling, and T. Dietterich. Transfer learning with an ensemble of background tasks. In *NIPS Workshop on Transfer Learning*, 2005.
25. N. Mehta, S. Ray, P. Tadepalli, and T. Dietterich. Automatic discovery and transfer of MAXQ hierarchies. In *International Conference on Machine Learning*, 2008.
26. L. Mihalkova, T. Huynh, and R. Mooney. Mapping and revising Markov Logic Networks for transfer learning. In *AAAI Conference on Artificial Intelligence*, 2007.
27. L. Mihalkova and R. Mooney. Transfer learning with Markov Logic Networks. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.
28. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
29. A. Niculescu-Mizil and R. Caruana. Inductive transfer for Bayesian network structure learning. In *Conference on AI and Statistics*, 2007.
30. S. Pan, J. Kwok, and Q. Yang. Transfer learning via dimensionality reduction. In *AAAI Conference on Artificial Intelligence*, 2008.
31. T. Perkins and D. Precup. Using options for knowledge transfer in reinforcement learning. Technical Report UM-CS-1999-034, University of Massachusetts, Amherst, 1999.
32. B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *International Conference on Machine Learning*, 1999.
33. R. Raina, A. Ng, and D. Koller. Constructing informative priors using transfer learning. In *International Conference on Machine Learning*, 2006.
34. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
35. M. Rosenstein, Z. Marx, L. Kaelbling, and T. Dietterich. To transfer or not to transfer. In *NIPS Workshop on Inductive Transfer*, 2005.
36. U. Ruckert and S. Kramer. Kernel-based inductive transfer. In *European Conference on Machine Learning*, 2008.
37. M. Sharma, M. Holmes, J. Santamaria, A. Irani, C. Isbell, and A. Ram. Transfer learning in real-time strategy games using hybrid CBR/RL. In *International Joint Conference on Artificial Intelligence*, 2007.
38. A. Sherstov and P. Stone. Action-space knowledge transfer in MDPs: Formalism, suboptimality bounds, and algorithms. In *Conference on Learning Theory*, 2005.
39. X. Shi, W. Fan, and J. Ren. Actively transfer domain knowledge. In *European Conference on Machine Learning*, 2008.
40. S. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3-4):323–339, 1992.
41. V. Soni and S. Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *AAAI Conference on Artificial Intelligence*, 2006.
42. D. Stracuzzi. Memory organization and knowledge transfer. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.
43. C. Sutton and A. McCallum. Composition of conditional random fields for transfer learning. In *Conference on Empirical methods in Natural Language Processing*, 2005.
44. R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

45. R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
46. E. Talvitie and S. Singh. An experts algorithm for transfer learning. In *International Joint Conference on Artificial Intelligence*, 2007.
47. F. Tanaka and M. Yamamura. Multitask reinforcement learning on the distribution of MDPs. *Transactions of the Institute of Electrical Engineers of Japan*, 123(5):1004–1011, 2003.
48. M. Taylor, N. Jong, and P. Stone. Transferring instances for model-based reinforcement learning. In *European Conference on Machine Learning*, 2008.
49. M. Taylor, G. Kuhlmann, and P. Stone. Accelerating search with transferred heuristics. In *ICAPS Workshop on AI Planning and Learning*, 2007.
50. M. Taylor, G. Kuhlmann, and P. Stone. Autonomous transfer for reinforcement learning. In *Conference on Autonomous Agents and Multi-Agent Systems*, 2008.
51. M. Taylor and P. Stone. Cross-domain transfer for reinforcement learning. In *International Conference on Machine Learning*, 2007.
52. M. Taylor and P. Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *Conference on Autonomous Agents and Multi-Agent Systems*, 2007.
53. M. Taylor, P. Stone, and Y. Liu. Value functions for RL-based behavior transfer: A comparative study. In *AAAI Conference on Artificial Intelligence*, 2005.
54. M. Taylor, S. Whiteson, and P. Stone. Transfer learning for policy search methods. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.
55. S. Thrun and T. Mitchell. Learning one more thing. In *International Joint Conference on Artificial Intelligence*, 1995.
56. L. Torrey, J. Shavlik, S. Natarajan, P. Kuppili, and T. Walker. Transfer in reinforcement learning via Markov Logic Networks. In *AAAI Workshop on Transfer Learning for Complex Tasks*, 2008.
57. L. Torrey, J. Shavlik, T. Walker, and R. Maclin. Relational skill transfer via advice taking. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.
58. L. Torrey, J. Shavlik, T. Walker, and R. Maclin. Relational macros for transfer in reinforcement learning. In *International Conference on Inductive Logic Programming*, 2007.
59. L. Torrey, T. Walker, J. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *European Conference on Machine Learning*, 2005.
60. T. Walsh, L. Li, and M. Littman. Transferring state abstractions between MDPs. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.
61. C. Wang and S. Mahadevan. Manifold alignment using Procrustes analysis. In *International Conference on Machine Learning*, 2008.
62. C. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.
63. A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *International Conference on Machine Learning*, 2007.